

# Cloud Development API Documentation

Garrett Haley(haleyg), Alea Weeks(weeksr),  
Daniel Domme(dommed), Ethan Patterson(patteret)

June 8, 2019

Contents

1	API Architecture Design	3
2	Data Schema	4
3	Reflection	4
3.1	Design . . . . .	4
3.2	Things we would like to change . . . . .	5

# 1 API Architecture Design

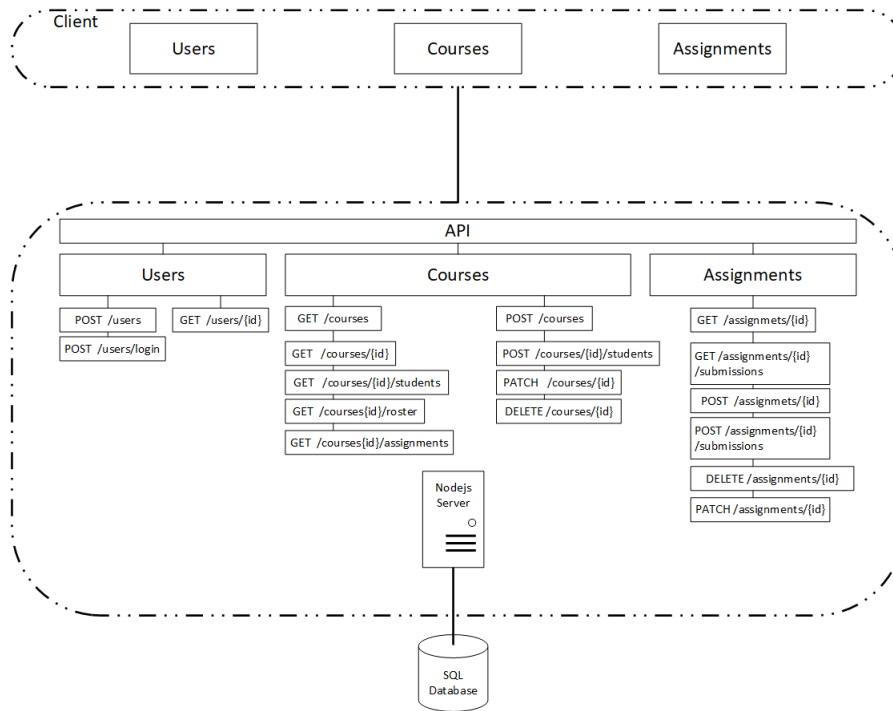


Figure 1: API Design

The above diagram is a pictorial example of the current design of our system. The client has access to three endpoint objects for requesting information. These endpoints are users, courses, and assignments. All endpoints are accessed through a single nodejs. All data is stored securely in an SQL server that only the nodejs has privileges to access.

## 1. Users endpoint

- POST /users # Adds a new user.
- POST /users/login # Allows a user to login.
- GET /users/{id} # Gets information for a specific user if authorized.

## 2. Courses endpoint

- GET /courses # Fetches a list of courses.
- GET /courses{id} # Fetch data for a specific course.
- GET /courses/{id}/students # Fetch data for a specific course.
- GET /courses{id}/roster # Fetch a text/cvs file listing students in a course.
- GET /courses/assignments # Fetches a list of assignments for a course.
- POST /courses # Create a new course.
- POST /courses/{id}/students # Update enrolment for a specific course.
- PATCH /courses/{id} # Update data for a specific course.
- DELETE /courses/{id} # Remove a specific course.

## 3. Assignments endpoint

- POST /assignments # Creates a new assignment.
- POST /assignments/{id}submissions # Creates a new submission for an assignment.
- GET /assignments{id} # Get information about a specific assignment.
- GET /assignments/{id}submissions # Fetch information about a specific assignment.
- PATCH /assignments/{id} # Update data for a specific assignment.
- DELETE /assignments/{id} # Remove a specific assignment.

## 2 Data Schema

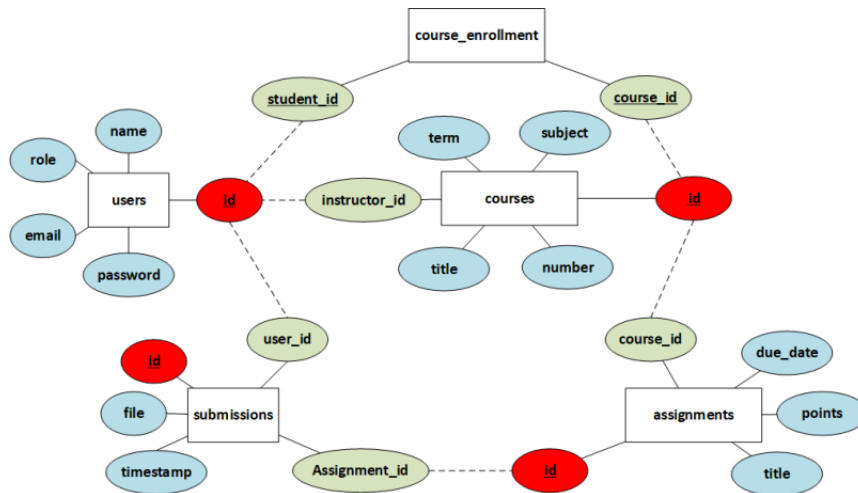






Figure 2: Data Schema

Figure 2 depicts the data schema currently in use by our SQL database.

1. White squares represent tables. 
2. Red elements represent a unique id for a table. 
3. Blue elements represent attributes of a table. 
4. Green elements represent foreign keys. All foreign keys have a dashed line connecting them to the location of their primary key. 
5. Elements with underscores also represent primary keys. This allows foreign keys to be primary.

## 3 Reflection

### 3.1 Design

We chose the API architecture shown in figure 1 as we thought this would be the fastest and most efficient way of getting data to and from our database to the user. Because Tarpaulin needs to provide features similar to Canvas, we wanted to make sure that all updates requested to the Tarpaulin service are immediately updated for all others clients. For this reason the speed at which information is propagated in our system is critical.

To achieve this goal our API server is implemented using nodejs. This allows our server to service multiple clients at a much faster rate than if it was implemented with a service designed using PHP. For information storage and retrieval we chose to implement our database using mysql. All resources suggested if speed is a concern, a SQL database would be the preferred option.

All request for information are first handled and processed by the single nodejs server. This server is then responsible for reaching out to the database to query for the requested information if the user meets all policies in place to access said information. Currently we have a rate limiter set for 5 attempts over a 6 second time period.

Currently admins have access to all information. Instructors can request less information than admins, but have far more access than a student. Instructors can currently post new assignments, get a list of student ids in a given class, add or remove students from a course, update a specific course, and request a course roster. The student role is the most restricted of the three. Users with the student role can submit assignments, and update assignments.

### 3.2 Things we would like to change

We followed the instructions listed in the provided openapi file for every endpoint specification. Because we followed the specifications in this file so closely this made our database less than normalized, as you can see in the data schema in figure 2. We believe that the roles in the user table could be split into smaller tables, this would make the database follow second normal form. With the current layout, it is difficult to manage a user who might have both the roles of student and instructor. Students can be teaching assistance for courses, this could cause issues with the current setup. Furthermore, this would allow us to only refer to user id and eliminate the confusing usage of student id and instructor id.

We would also like to have incorporated more security for our server. Currently to obtain a login token a user must send their password in plain text to the server. This leaves our users, and admins, vulnerable to man in the middle attacks and other eavesdropping attacks.

We would also like to implement endpoints for downloading files that have been submitted for a specific assignment. Currently there is no way for a student to download an assignment that they have submitted. We believe by adding this feature students would feel reassured they submitted the correct file. This also adds the ability of being able to access a submitted assignment from a different machine.

Currently we store all assignments into a single folder. In the future it would be nice to generate a parent folder for a course and fill it with assignment folders for said course. This would keep items organized in a way that any information about previous courses could be easily collected and referenced.

#### Conclusion

Using mysql for our database was the correct decision. Our entire team was familiar and comfortable using mysql. This allowed us to move quickly in implementing the database. The thing we wish we could change the most is the relationship between tables. Currently they are not in normal form. If we could do this assignment over again with more control on data relations. This what we would change.